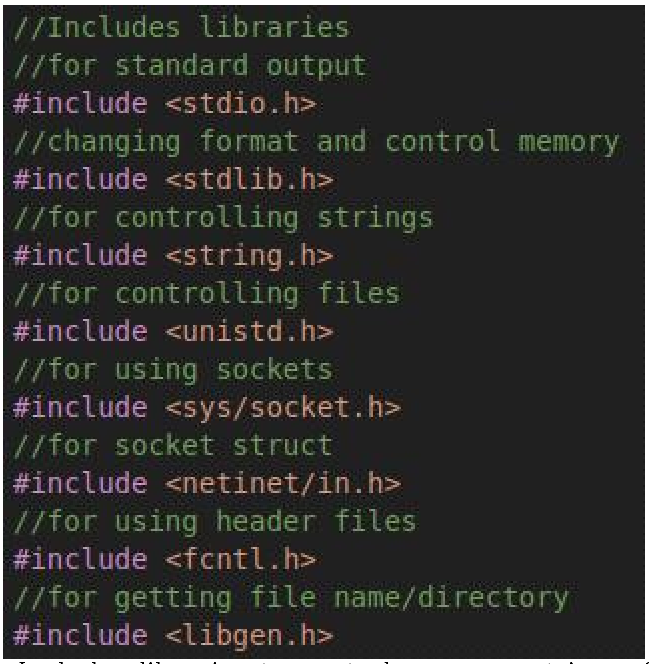
Computer Network Project

Implementing Web-Server

과목명	Computer Networks	**학과**	컴퓨터학부
교수명	이석복	**학번**	2020052551
제출일	2024.04.13	**이름**	성주원

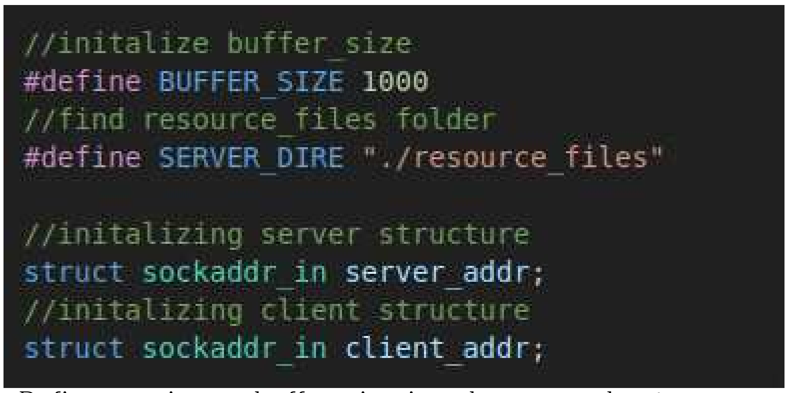
한양대학교 ERICA
Education Research Industry Cluster @ Ansan

1. Give a high-level description of your server's design



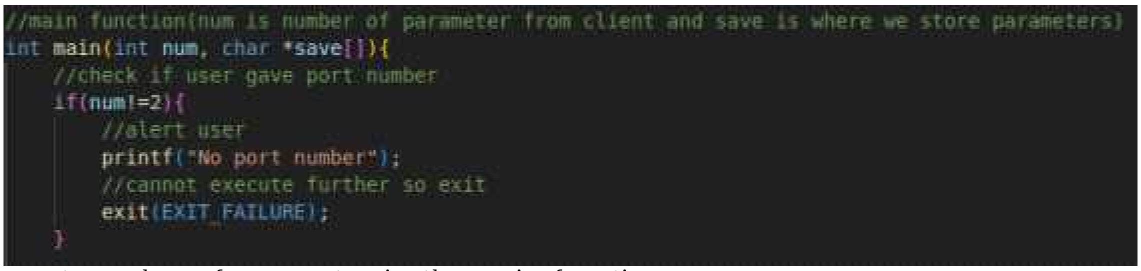
```c
//Includes libraries
//for standard output
#include <stdio.h>
//changing format and control memory
#include <stdlib.h>
//for controlling strings
#include <string.h>
//for controlling files
#include <unistd.h>
//for using sockets
#include <sys/socket.h>
//for socket struct
#include <netinet/in.h>
//for using header files
#include <fcntl.h>
//for getting file name/directory
#include <libgen.h>
```

-Includes libraries to control memory, strings, files, and header
Also to use socket



```c
//initalize buffer_size
#define BUFFER_SIZE 1000
//find resource_files folder
#define SERVER_DIRE "./resource_files"

//initalizing server structure
struct sockaddr_in server_addr;
//initalizing client structure
struct sockaddr_in client_addr;
```

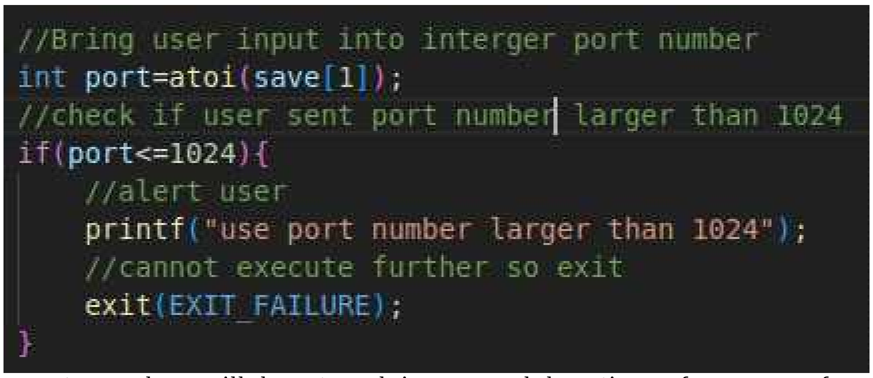
-Define maximum buffer size in advance and set resource file directory
-To establish socket, needs to initialize server and client structure



```c
//main function(num is number of parameter from client and save is where we store parameters)
int main(int num, char *save[]){
    //check if user gave port number
    if(num!=2){
        //alert user
        printf("No port number");
        //cannot execute further so exit
        exit(EXIT_FAILURE);
    }
```

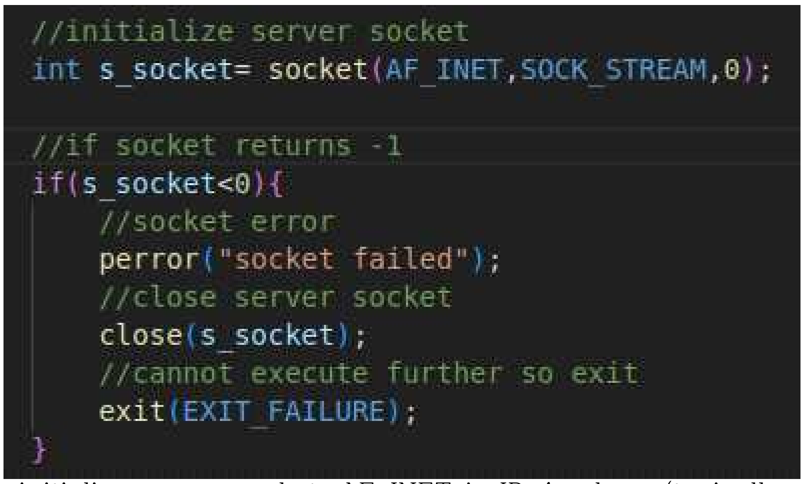
- get number of parameter in the main function
ㄴif server didn't give port number (ex. ./myserver), parameter includes just 1 so

let server know there is no port number included. If port number is not given, just exit



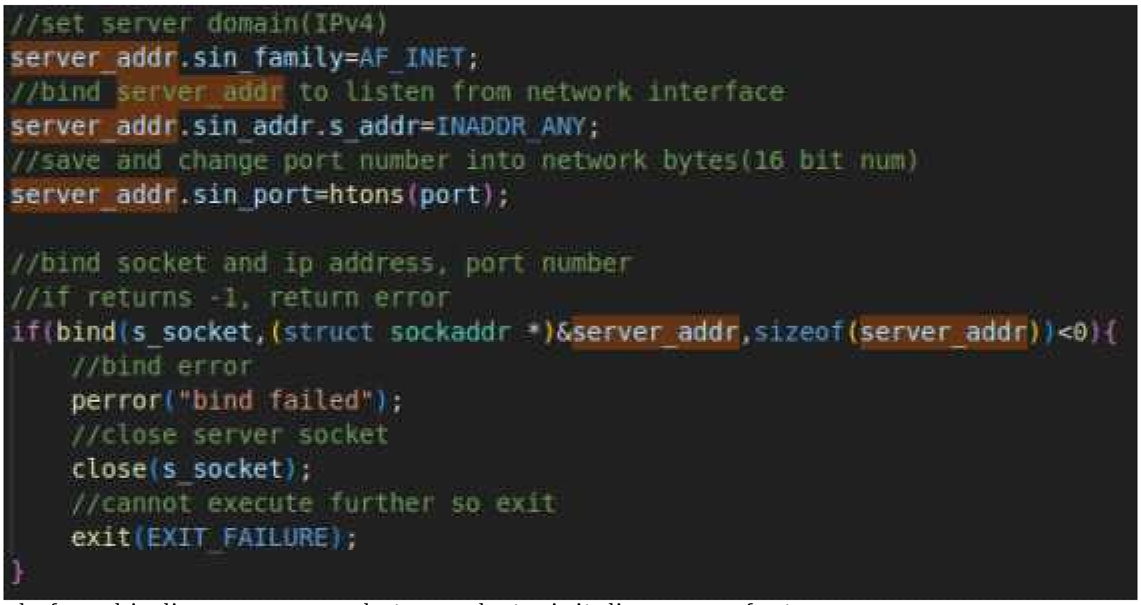
```
//Bring user input into interger port number
int port=atoi(save[1]);
//check if user sent port number larger than 1024
if(port<=1024){
    //alert user
    printf("use port number larger than 1024");
    //cannot execute further so exit
    exit(EXIT_FAILURE);
}
```

-port number will be stored in second location of save so for readability, store in integer port
-if port is in between 0~1024(which is not recommended port number), alert server to use port number larger than 1024.
-exit if port number is in between 0~1024



```
//initialize server socket
int s_socket= socket(AF_INET,SOCK_STREAM,0);

//if socket returns -1
if(s_socket<0){
    //socket error
    perror("socket failed");
    //close server socket
    close(s_socket);
    //cannot execute further so exit
    exit(EXIT_FAILURE);
}
```

-initialize server socket, AF_INET is IPv4 scheme(typically used)
-SOCK_STREAM is using TCP protocol which is also typically used
-to let socket to select protocol automatically, use 0(typically used)
-socket returns -1 if failed so if s_socket returns value less than 0, it means something has gone wrong
ㄴalert server that socket failed and since socket needs to be closed before program ends, close socket and exit program



```
//set server domain(IPv4)
server_addr.sin_family=AF_INET;
//bind server_addr to listen from network interface
server_addr.sin_addr.s_addr=INADDR_ANY;
//save and change port number into network bytes(16 bit num)
server_addr.sin_port=htons(port);

//bind socket and ip address, port number
//if returns -1, return error
if(bind(s_socket,(struct sockaddr *)&server_addr,sizeof(server_addr))<0){
    //bind error
    perror("bind failed");
    //close server socket
    close(s_socket);
    //cannot execute further so exit
    exit(EXIT_FAILURE);
}
```

-before binding server socket, needs to initalize some features

ㄴuse IPv4 using AF_INET

ㄴto listen everything from incoming request, use INADDR_ANY

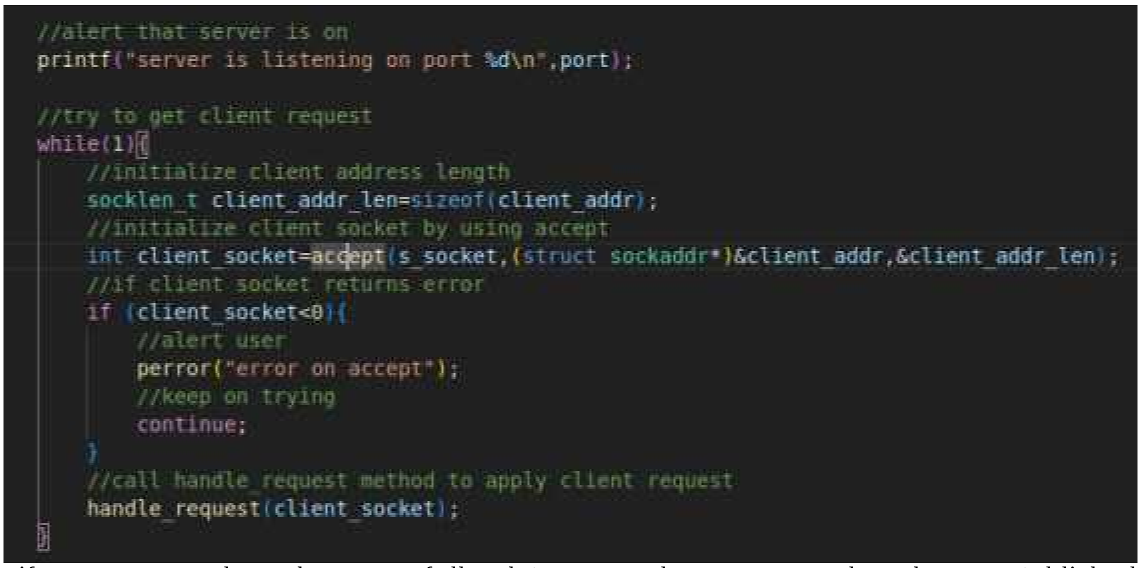
ㄴto initialize port, use hton command to parse into network bytes

-bind server socket and if returns less than 0, something has gone wrong so return error

ㄴalert server bind has failed

ㄴclose socket before ending program

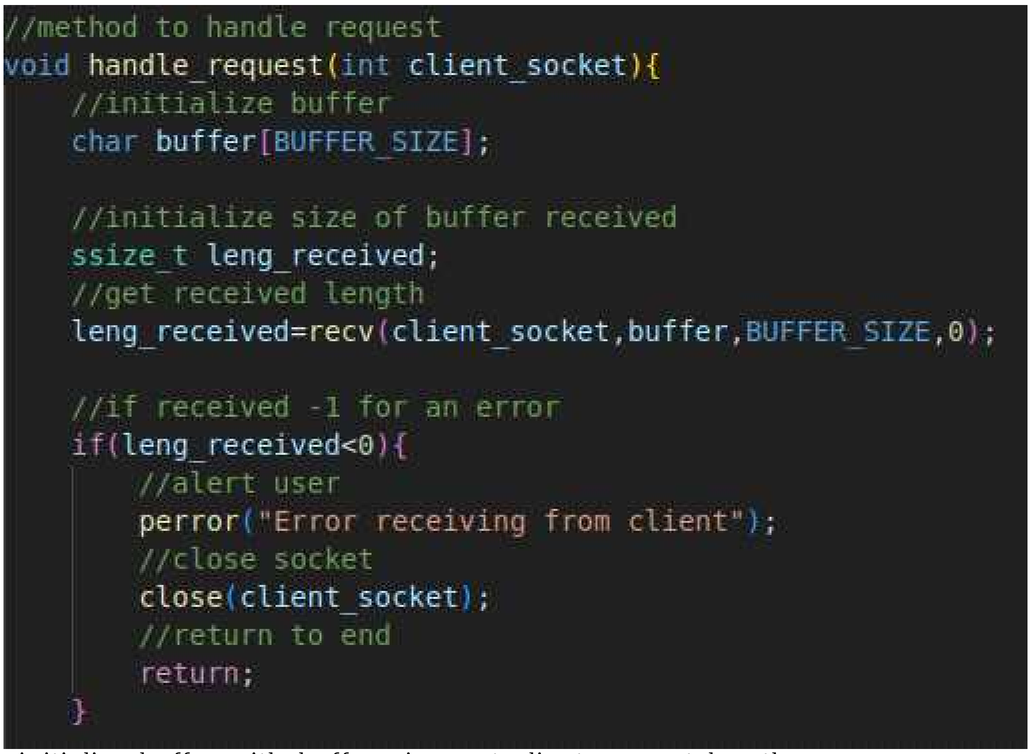
ㄴend program



```
//alert that server is on
printf("server is listening on port %d\n",port);

//try to get client request
while(1){
    //initialize client address length
    socklen_t client_addr_len=sizeof(client_addr);
    //initialize client socket by using accept
    int client_socket=accept(s_socket,(struct sockaddr*)&client_addr,&client_addr_len);
    //if client socket returns error
    if (client_socket<0){
        //alert user
        perror("error on accept");
        //keep on trying
        continue;
    }
    //call handle_request method to apply client request
    handle_request(client_socket);
}
```

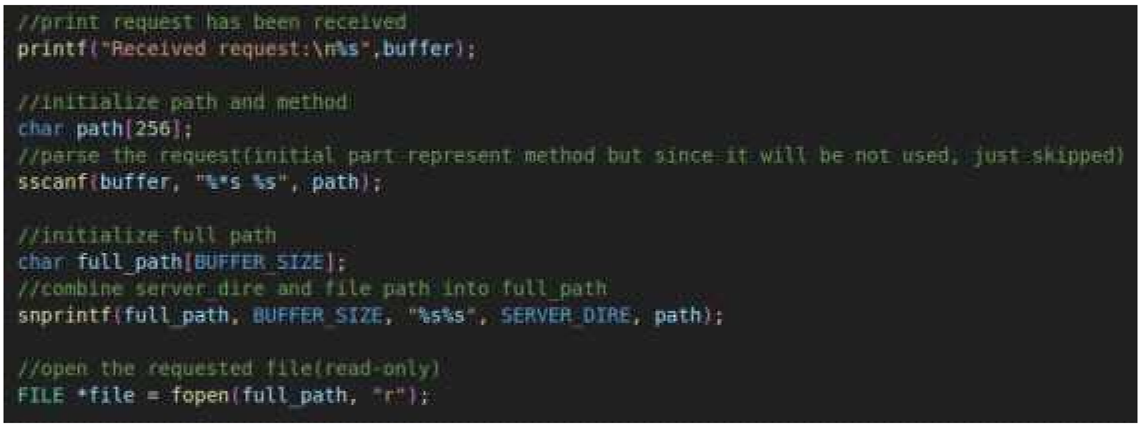
-if server was bound successfully, let server know server has been established successfully and ready to listen from client

-until program is not closed, keep listening from client
ㄴinitialize client address size
ㄴinitialize client socket and keep listening for client request
ㄴif accept command gone something wrong, alert server something has gone wrong on accept command
-go to handle request method to execute client request



```
//method to handle request
void handle_request(int client_socket){
    //initialize buffer
    char buffer[BUFFER_SIZE];

    //initialize size of buffer received
    ssize_t leng_received;
    //get received length
    leng_received=recv(client_socket,buffer,BUFFER_SIZE,0);

    //if received -1 for an error
    if(leng_received<0){
        //alert user
        perror("Error receiving from client");
        //close socket
        close(client_socket);
        //return to end
        return;
    }
```

-initialize buffer with buffer_size, get client request length
ㄴif client request length is less than 0, which means there is no data or something has gone wrong, alert server something has gone wrong. Before closing program, close client socket and return to main method and close



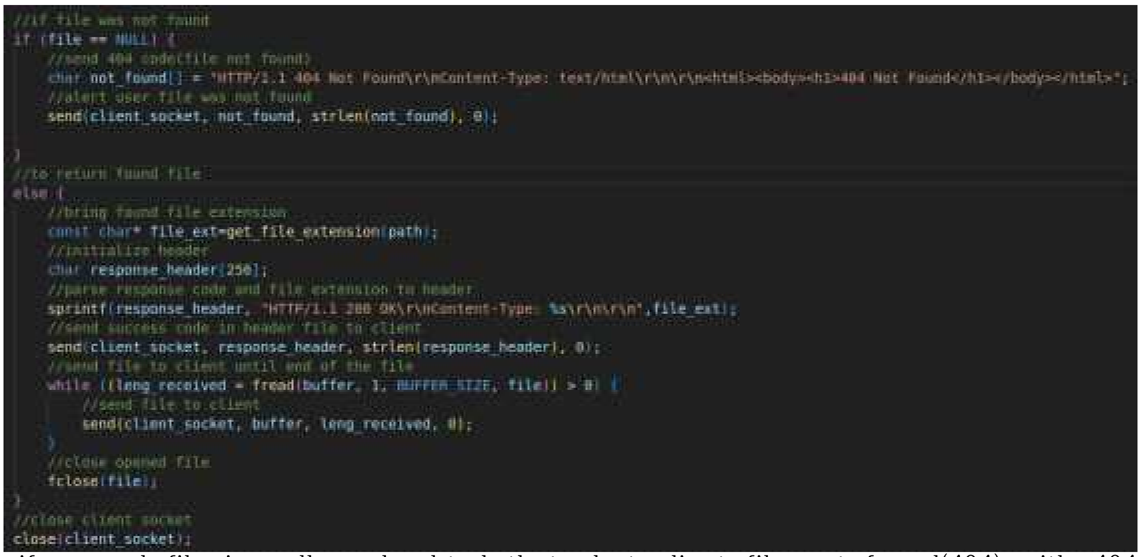
```
//print request has been received
printf("Received request:\n%s",buffer);

//initialize path and method
char path[256];
//parse the request(initial part represent method but since it will be not used, just skipped)
sscanf(buffer, "%*s %s", path);

//initialize full path
char full_path[BUFFER_SIZE];
//combine server dire and file path into full_path
snprintf(full_path, BUFFER_SIZE, "%s%s", SERVER_DIRE, path);

//open the requested file(read-only)
FILE *file = fopen(full_path, "r");
```

-Let server know request has been successfully received

-get file name from buffer

-combine SERVER_DIRE and path into full_path to get proper address of file

-try to open file



```
//if file was not found
if (file == NULL) {
    //send 404 code(file not found)
    char not_found[] = "HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n<html><body><h1>404 Not Found</h1></body></html>";
    //alert user file was not found
    send(client_socket, not_found, strlen(not_found), 0);

}
//to return found file
else {
    //bring found file extension
    const char* file_ext=get_file_extension(path);
    //initialize header
    char response_header[256];
    //parse response code and file extension to header
    sprintf(response_header, "HTTP/1.1 200 OK\r\nContent-Type: %s\r\n\r\n",file_ext);
    //send success code in header file to client
    send(client_socket, response_header, strlen(response_header), 0);
    //send file to client until end of the file
    while ((leng_received = fread(buffer, 1, BUFFER_SIZE, file)) > 0) {
        //send file to client
        send(client_socket, buffer, leng_received, 0);
    }
    //close opened file
    fclose(file);
}
//close client socket
close(client_socket);
```

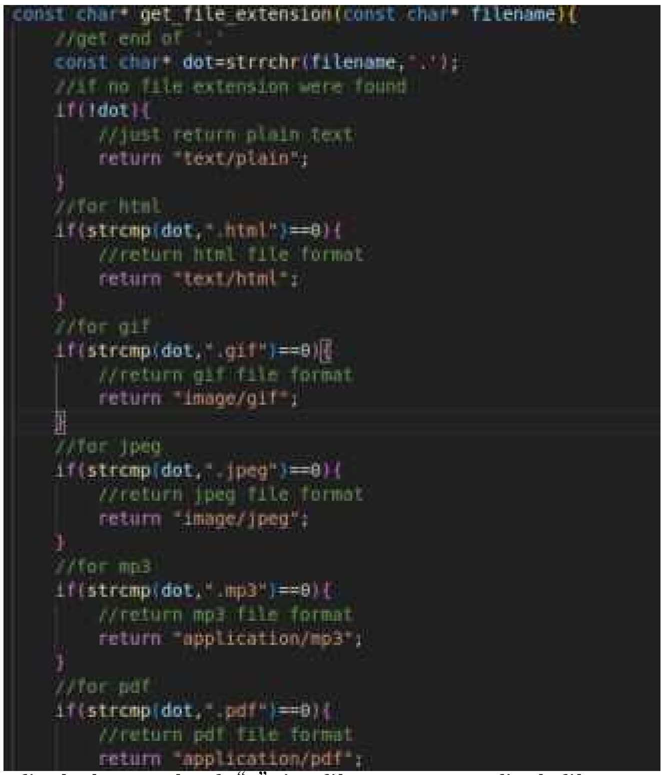
-if opened file is null, make html that alert client file not found(404) with 404 error code and send to client socket

-if file was successfully found, get file extension from path using get_file extension(explained later)

ㄴparse file and file extension and send to client

ㄴuntil end of the file, keep send to client

-if file is all sent, close opened file and close client_socket

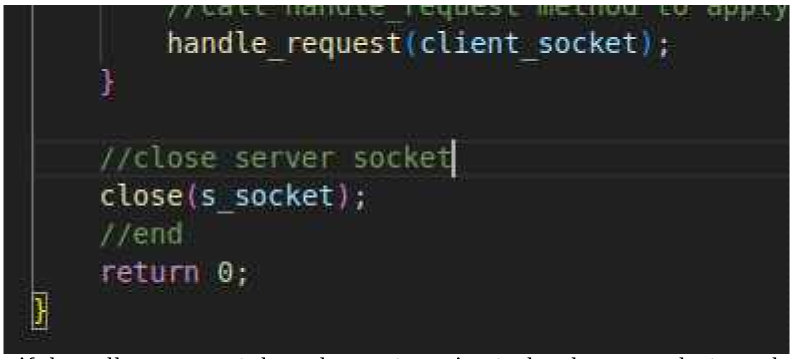


```
const char* get_file_extension(const char* filename){
    //get end of '.'
    const char* dot=strrchr(filename,'.');
    //if no file extension were found
    if(!dot){
        //just return plain text
        return "text/plain";
    }
    //for html
    if(strcmp(dot,".html")==0){
        //return html file format
        return "text/html";
    }
    //for gif
    if(strcmp(dot,".gif")==0){
        //return gif file format
        return "image/gif";
    }
    //for jpeg
    if(strcmp(dot,".jpeg")==0){
        //return jpeg file format
        return "image/jpeg";
    }
    //for mp3
    if(strcmp(dot,".mp3")==0){
        //return mp3 file format
        return "application/mp3";
    }
    //for pdf
    if(strcmp(dot,".pdf")==0){
        //return pdf file format
        return "application/pdf";
```

-find the end of "." in file name to find file extension

-return appropriate format for each file extension

(Back to main method)



```
            //call handle_request method to apply
            handle_request(client_socket);
        }

        //close server socket
        close(s_socket);
        //end
        return 0;
}
```

-if handle_request has been terminated, close socket and end program

2. What difficulties did you face and how did you solve them?

-First of all, since I wasn't used to using network api so I didn't know where to start. But there were a lot of sources that implemented web server using c so I decided to refer them.

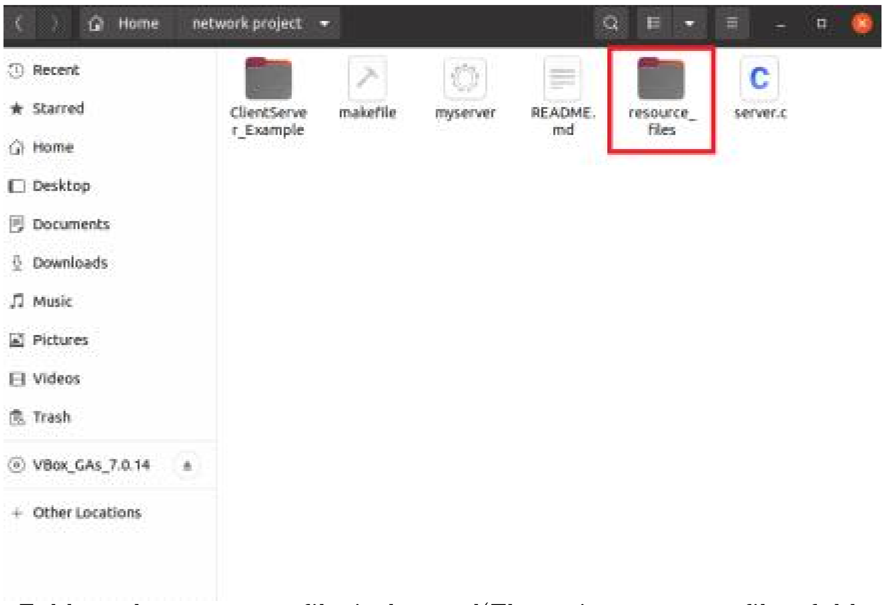
-Also I had difficult time using network api because for example, when using a bind command, it has structure that has format that needs to be initialized before using bind. So I had to look into how to make structure according to bind command. Other than bind, I also had hard time using accept, initializing socket, etc.

-I had to look into file format for each files. For example, I realized that I had to get file extension before sending response code in advance to let client web know what type of file will be sent. So I made method that extract file extension and returns file type into format that client socket would interpret

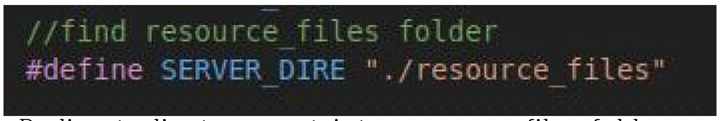
-There were much more things that I had hard time to successfully implement web server, but those were the main problems that I had.

3. Include and briefly explain some sample outputs of your client-server (e.g. in Part A you should be able to see an HTTP request)

-For readability, I made resource_files folder and put all the resources to test my web server.

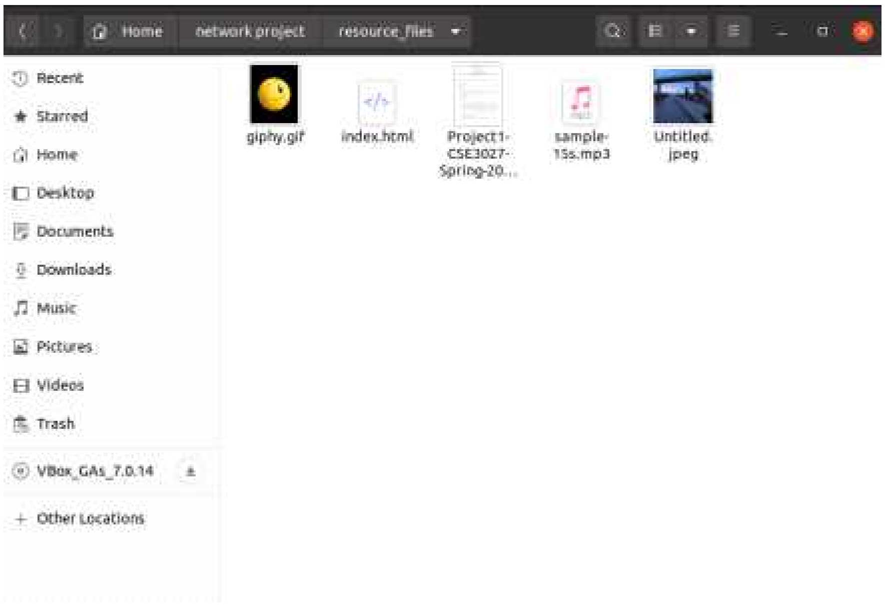


-Folder where server file is located(There is resource_files folder)

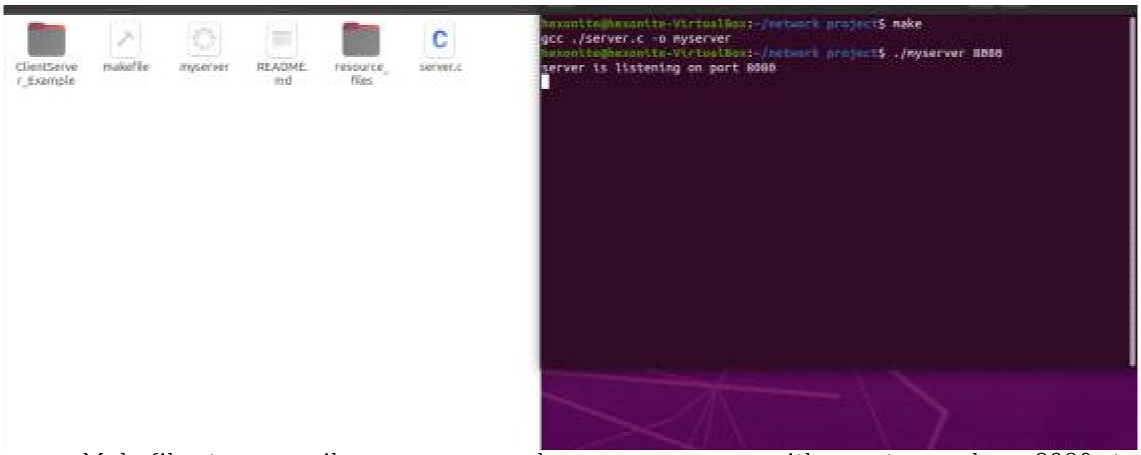


```
//find resource_files folder
#define SERVER_DIRE "./resource_files"
```

-Redirect client request into resource_files folder

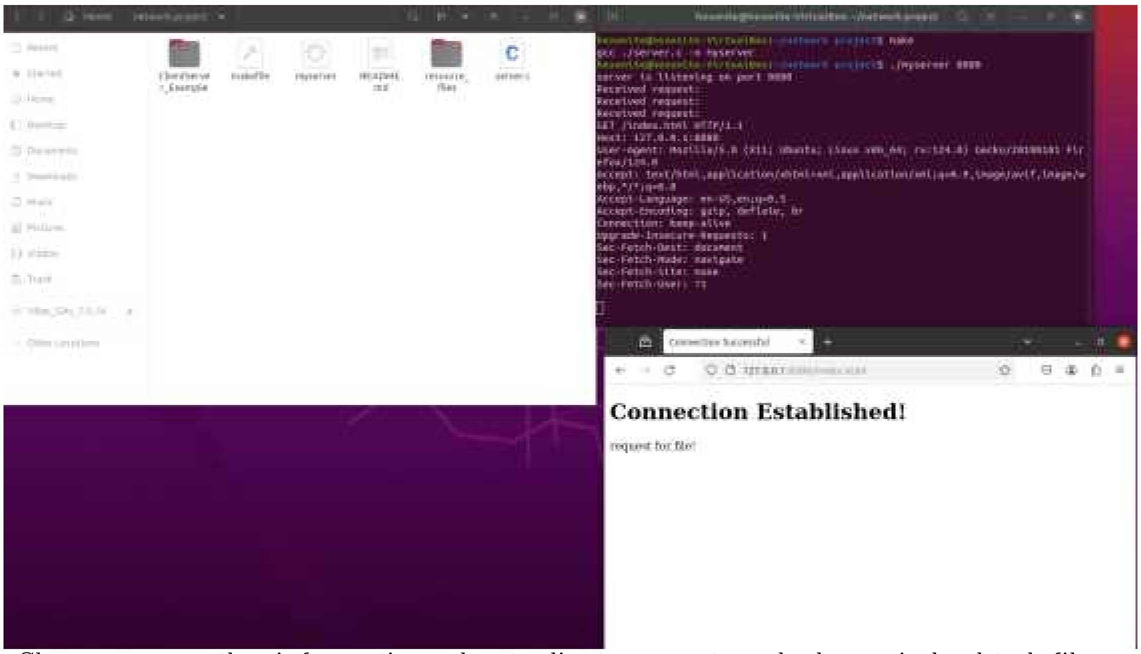


-Various files located to test web server(gif, html, pdf, mp3, jpeg)



```
hexonite@hexonite-VirtualBox:~/network project$ make
gcc ./server.c -o myserver
hexonite@hexonite-VirtualBox:~/network project$ ./myserver 8080
server is listening on port 8080
```

-use Makefile to compile server.c and use myserver with port number 8080 to establish web server

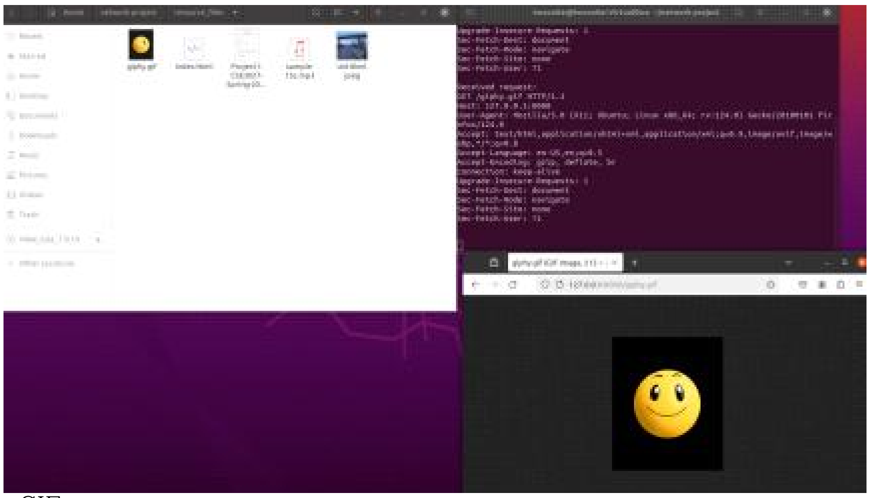


```
hexonite@hexonite-VirtualBox:~/network project$ make
gcc ./server.c -o myserver
hexonite@hexonite-VirtualBox:~/network project$ ./myserver 8080
server is listening on port 8080
Received request:
Received request:
Received request:
GET /index.html HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

Connection Established!

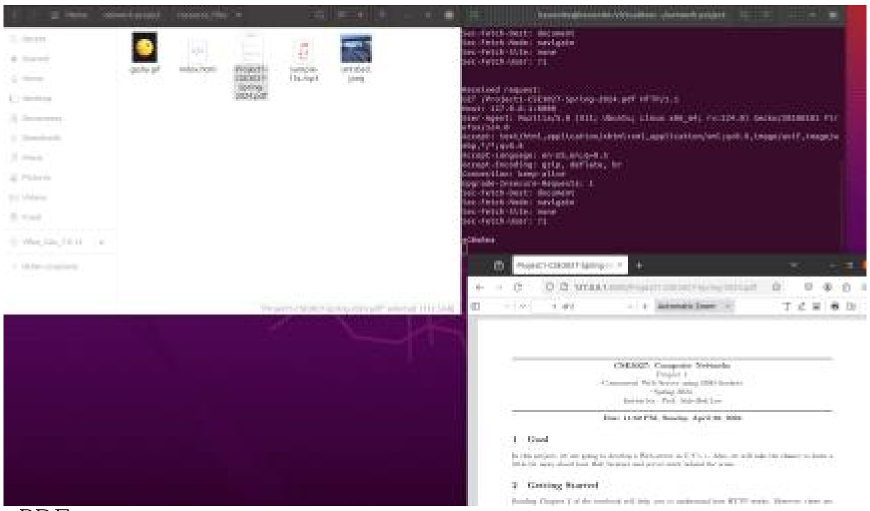
request for file!

-Shows server the information about client request and shows index.html file as requested
-Not only html file but other types of format works too



```
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

Received request:
GET /giphy.gif HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Fir
efox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

-GIF



```
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

Received request:
GET /Project1-CSE3027-Spring-2024.pdf HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Fir
efox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

[illegible]
```

CSE3027: Computer Networks
Project 1
Concurrent Web Server using BSD Sockets
Spring 2024

Due: 11:59 PM, Sunday, April 28, 2024

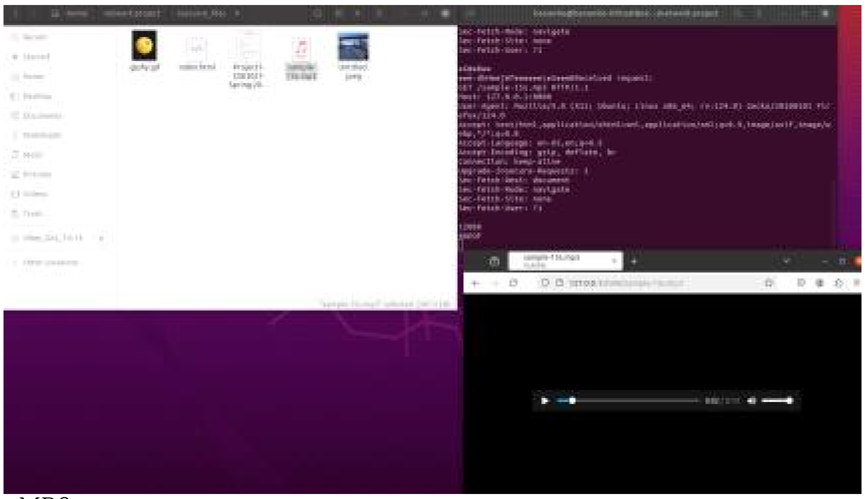
1 Goal

[illegible]

2 Getting Started

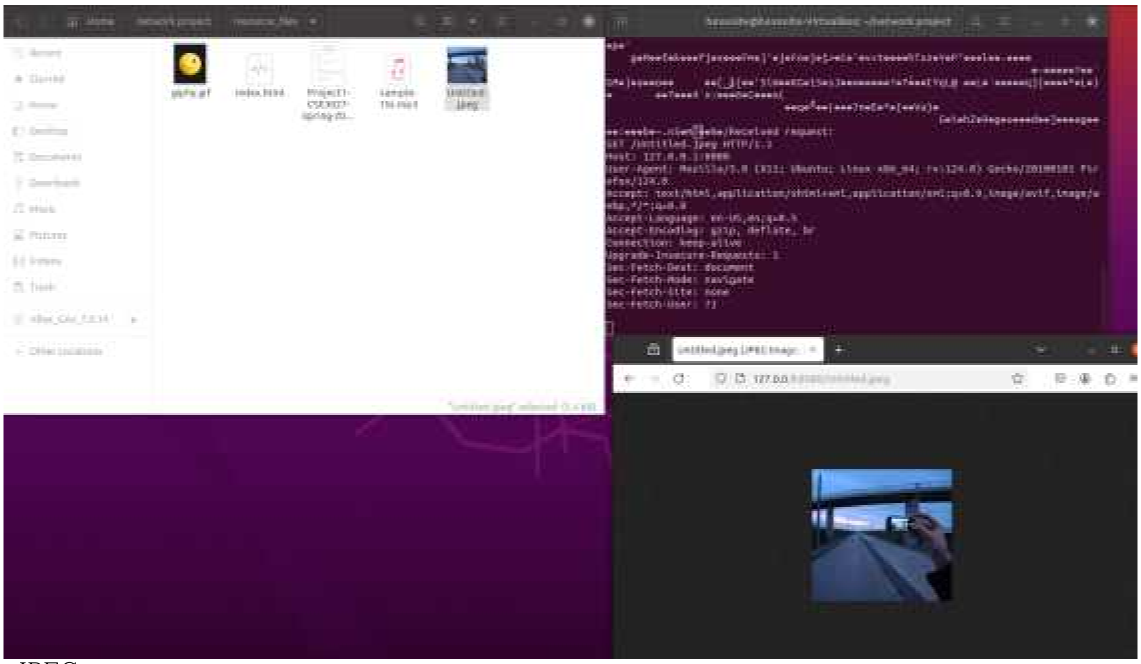
[illegible]

-PDF



```
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

[illegible]
[illegible]Received request:
GET /sample-1 1s.mp3 HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Fir
efox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

[illegible]
[illegible]
```

-MP3



```
[illegible]
[illegible]Received request:
GET /untitled.jpeg HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

-JPEG